REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* AUG 2011	**2. REPORT TYPE** **Conference Paper** (Post Print)	**3. DATES COVERED** *(From - To)* APR 2010 – Dec 2010

4. TITLE AND SUBTITLE EVALUATING QOS-ENABLED INFORMATION MANAGEMENT SERVICES IN A NAVY OPERATIONAL CONTEXT	**5a. CONTRACT NUMBER** FA8750-08-C-0022
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 62788F
6. AUTHOR(S) Asher D. Sinclair (AFRL) Aaron M. Paulos; Joseph P. Loyall (BBN)	**5d. PROJECT NUMBER** 558J
	5e. TASK NUMBER 08
	5f. WORK UNIT NUMBER 01
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) BBN Technologies 10 Moulton Street Cambridge, MA 02138	**8. PERFORMING ORGANIZATION REPORT NUMBER** N/A
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/RISE 525 Brooks Road Rome NY 13441-4505	**10. SPONSOR/MONITOR'S ACRONYM(S)** N/A
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-RI-RS-TP-2011-28

12. DISTRIBUTION AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED. PA #: 88ABW-2011-1780
DATE CLEARED: 29 MAR 2011

13. SUPPLEMENTARY NOTES
© 2011 SPIE. Publication in Defense Transformation and Net-Centric Systems 2011, Proc. of SPIE Vol 8062, 806203-1. This work is copyrighted. One or more of the authors is a U.S. Government employee working within the scope of their Government job; therefore, the U.S. Government is joint owner of the work and has the right to copy, distribute, and use the work. All other rights are reserved by the copyright owner.

14. ABSTRACT Information Management (IM) services support the discovery, brokering, and dissemination of mission-critical information based on the information's content and characteristics. IM services support the dissemination of future information (through subscriptions) and past information (through queries) regardless of its source. To be useful across enterprise and tactical environments. IM services need mission-driven Quality of Service (QoS) features as part of their core functionality. We have developed QoS management features, QoS Enabled Dissemination (QED), that extend an Air Force Research Laboratory (AFRL) developed set of IM services, Phoenix. The paper describes the results of a joint services experiment evaluating QED and Phoenix in a US Navy scenario involving multiple ships connected by a Disconnected, Intermittent, Limited (DIL) satellite network. The experiments evaluate QED and Phoenix's ability to (1) provide IM in the Wide Area Network (WAN) context of the satellite communications, which includes long latencies and background traffic not under QED control; (2) control and utilize active-precedence and queue management features provided by the WAN; (3) handle severe network overload, network disruptions and dynamic changes in policies.

15. SUBJECT TERMS
Quality of Service, Information management services, DIL networks, message prioritization, policy

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON** ASHER D. SINCLAIR
a. REPORT U	**b. ABSTRACT** U	**c. THIS PAGE** U	UU	24	**19b. TELEPHONE NUMBER** (*Include area code*) N/A

Evaluating QoS-Enabled Information Management Services in a Navy Operational Context

Aaron M Paulos[a], Asher Sinclair[b] and Joseph P. Loyall[a]
[a]BBN Technologies, 10 Moulton Street, Cambridge, MA 02138
[b]US Air Force Research Laboratory, 525 Brooks Rd, Rome, NY 13441

ABSTRACT

Information Management (IM) services support the discovery, brokering, and dissemination of mission-critical information based on the information's content and characteristics. IM services support the dissemination of future information (through subscriptions) and past information (through queries) regardless of its source. To be useful across enterprise and tactical environments, IM services need mission-driven Quality of Service (QoS) features as part of their core functionality. We have developed QoS management features, QoS Enabled Dissemination (QED), that extend an Air Force Research Laboratory (AFRL) developed set of IM services, Phoenix. This paper describes the results of a joint services experiment evaluating QED and Phoenix in a US Navy scenario involving multiple ships connected by a Disconnected, Intermittent, Limited (DIL) satellite network. Experiments evaluate QED and Phoenix's ability to (1) provide IM in the Wide Area Network (WAN) context of the satellite communications, which includes long latencies and background traffic not under QED control; (2) control and utilize active-precedence and queue management features provided by the WAN; (3) handle severe network overload, network disruptions, and dynamic changes in policies; and (4) successfully enforce deadlines and information replacement policies.
Keywords: Quality of service, information management services, DIL networks, message prioritization, policy

1. INTRODUCTION

Bridging Information Management (IM) services from enterprise to tactical environments requires Quality of Service (QoS) capabilities to effectively utilize constrained and shared resources in service of mission objectives and to handle dynamic and hostile conditions. As part of ongoing research, we have developed QoS Enabled Dissemination (QED) features [10] for a set of US Air Force developed information management (IM) services, *Phoenix* [5].

Phoenix is a set of active IM services developed by the US Air Force Research Laboratory (AFRL). The core concept of Phoenix is an active information management model where clients are information publishers and consumers communicate anonymously with other clients via shared IM services, such as publication, discovery, brokering, archiving, and querying [1], [5]. Information is published in the form of typed information objects (IOs) consisting of payload and indexable metadata describing the object and its payload. Consumers make requests for future (subscription) or past (query) information using predicates, e.g., via XPath [21] or XQuery [22], over IO types and metadata values. The core set of Phoenix IM services includes the following services, shown in Figure 1:

- *Submission* service receives IOs published by clients.
- *Broker* service matches the type and metadata of IOs to the type and query expression of subscriptions registered by information subscribers.
- *Archive* service inserts IOs into a persistent IO data store.
- *Query* service processes queries consisting of predicates over the metadata of archived IOs.
- *Dissemination* service sends query results to querying clients and IOs that match subscription predicates to subscribing clients.

QED provides a set of policy-driven QoS managers and mechanisms that ensure the timely brokering and dissemination of important information through the IM services, ensure smoothness of information brokering and dissemination, make tradeoffs to favor completeness or fidelity when systems are overloaded, and enforce client preferences and priorities. QED extends the Phoenix IM services with QoS management services, including a *Dissemination Service* that performs

This research has been sponsored by the United States Air Force Research Laboratory under contract FA8750-08-C-0022.

prioritization of information, bandwidth management, and information shaping, and a *Task Manager* that controls the CPU or Virtual Machine threads that schedule brokering and other CPU intensive tasks. These *local* QoS managers (local because they manage QoS at a single control point) are under the control of an *Aggregate QoS Manager* that stores, selects, and distributes policies to achieve end-to-end, consistent QoS for multiple publish-subscribe information flows.

We have previously described QED's architecture, prototype implementation, and mechanisms and services [9] [10] [11] [14]. To provide QoS in IM systems, the QED prototype, shown in Figure 2, performs the following:

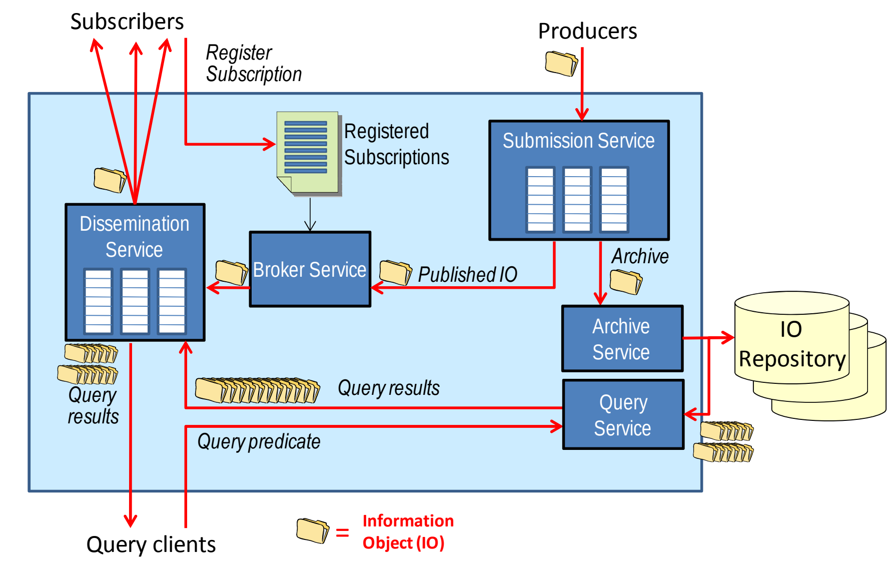

Figure 1. Phoenix core information management services.

- Manages the scheduling of threads and bandwidth typically hidden behind service interfaces.
- Provides and enforces QoS policy that is specified at a high level, based on user, information, and system concepts.
- Provides policy-driven aggregate QoS management across control points and across users, despite the publishing and consuming clients being decoupled from one another.
- Handles information that varies in size and processing that varies in time.
- Separately schedules the brokering of requests and dissemination of information, so that fanout is managed and important requests are brokered before less important requests. Policy driven scheduling expedites the delivery of information to important consumers in preference to delivery to less important consumers.

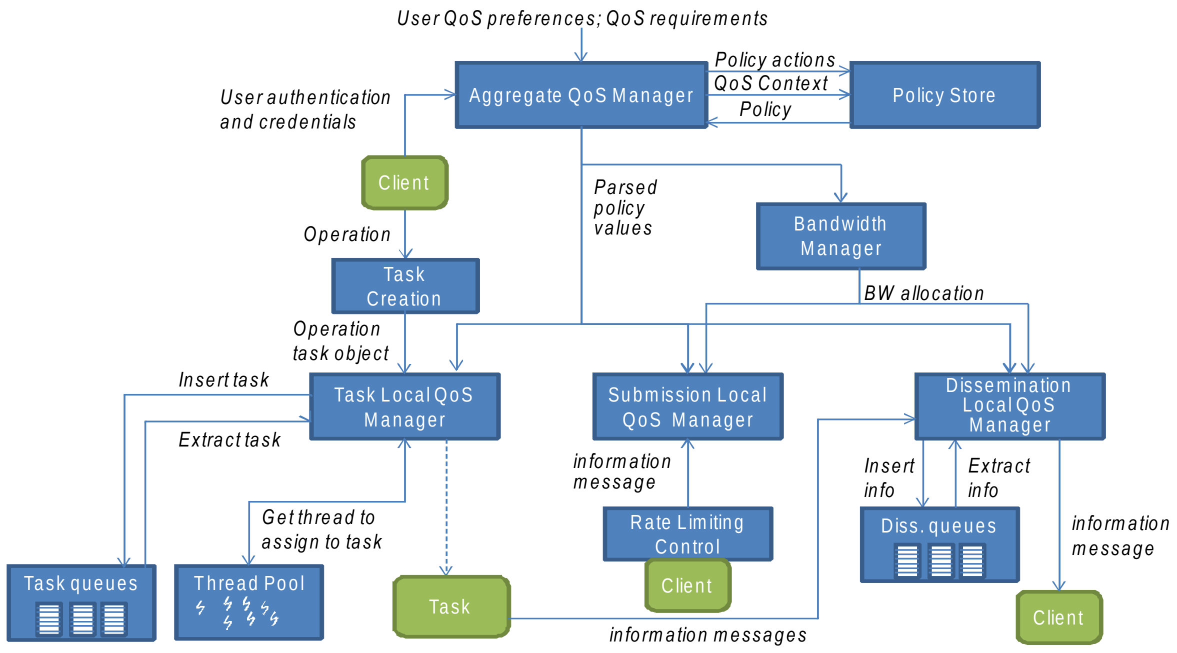

Figure 2. QED provides QoS management capabilities for IM Services.

As documented in our previous publications, we have evaluated QED's ability to allocate bandwidth, enforce QoS policies, manage conflicting demands on resources, and enforce client preferences for deadlines, replacement of stale information, and information format. We have evaluated these both qualitatively through operationally-relevant demonstrations and quantitatively, through focused sets of laboratory experiments.

In this paper, we describe the application of Phoenix IM services enhanced with QED's QoS management features (Phoenix+QED) to the needs of Navy shipboard operations, specifically the communication of information between Combat Systems (CS) and Command and Control (C2) Systems over a DIL network, i.e., the Navy's Extremely High Frequency Time Division Multiple Access (TDMA) Interface Processor (EHF-TIP) [24]. These capabilities are evaluated in a Limited Technology Experiment (LTE) conducted by the US Navy. The LTE consisted of a multi-ship environment across a Disconnected, Intermittent, Limited (DIL) satellite Wide Area Network (WAN) [24]. One of the LTE's defined *Mission Threads* (i.e., sets of experimental scenarios for a specific configuration of technologies) evaluated Phoenix+QED's ability to move information from one Naval platform to another through the DIL WAN with management of limited bandwidth, prioritization of information based on mission policies, disruption tolerance, resilience to overload, priority shifts, and enforcement of deadlines and replacement policies.

The remainder of this paper is structured as follows. Section 2 describes Phoenix+QED's participation in the Navy's CS/C2 LTE and the challenges associated with applying Phoenix and QED to the Navy environment. Section 3 describes the experiments run as part of the Navy LTE and the metrics collected. Section 4 provides the results of the experiments with Phoenix+QED in the Navy LTE. Section 5 describes related work. Finally, Section 6 provides some lessons learned and concluding remarks.

2. PHOENIX+QED IN A NAVAL SCENARIO

In early 2010 the coordinators of the US Navy CS/C2 LTE specified goals for evaluating the following three targets:

- Capabilities offering data transparency between Combat Systems and Command and Control Systems,
- Cross-platform resource discovery and federation capabilities between a simulated Aircraft Carrier (CVN) and two US Navy Destroyer (DDG) platforms, and
- Third-party IM techniques for disseminating track and readiness data over a DIL satellite network.

Over subsequent months, five experiment *Mission Threads* from various organizations addressed specific tasks related to one or more of the overarching LTE goals. Each mission thread evaluated different technologies in the Navy context and their ability to meet some or all of the LTE goals. An objective experimentation team from the Navy elicited requirements, structured experiments, conducted execution, and analyzed results of the five mission threads. This paper describes the challenges and results of integrating and using QED-enhanced Phoenix (Phoenix+QED) as an IM Service in *Mission Thread 3* of the Navy LTE.

Quantitatively, Mission Thread Three evaluates Phoenix+QED's ability to function as an edge-of-the-WAN IM service. In this role, Phoenix+QED is responsible for inspecting System Track and Readiness data, enforcing mission priorities over a select set of derived information, and managing the brokering and dissemination of information from local C2 Systems, over a shared and resource constrained SATCOM link, to remote C2 subscribers. In Section 3, we describe the set of experiments that were conducted and the metrics that were collected to assess Phoenix+QED's effectiveness as an edge service. The results described in Section 4 show that QED effectively manages load and disruption between the platforms and prioritizes information delivery to reduce latency and improve throughput of important information through the DIL network.

Figure 3 shows the architecture of the CS/C2 LTE and the Phoenix+QED Mission Thread 3. It is a full mesh network, with track and readiness information generated on each platform disseminated to the other two platforms. Locally produced track and readiness information is made available via a local Enterprise Service Bus (ESB). A Phoenix client (*Proxy*) receives Naval data from the ESB and encapsulates it in a structured IO, which the QED-enhanced Dissemination Service (DS) disseminates to the other platforms. The DS performs all QoS management, prioritizing outgoing information, replacing old tracks, enforcing deadlines, and enforcing bandwidth limits. Phoenix *Channels* handle the delivery of IOs to the transport layer and the receipt on the other side. An *SSL-TOS Channel* developed for the LTE sets the appropriate Secure Socket Layer (SSL) and Type of Service (TOS) parameters to utilize encryption and network-level packet prioritization (i.e., differential type of service code points), respectively. A Phoenix subscriber client (*Sub*) receives the disseminated IOs on the receiving side, strips off the IO-specific structure and pushes the resulting track or readiness data (in *Plain Old Java Object, POJO,* format) onto the receiving platform's ESB.

The rest of this section describes the challenges, details, and our experiences in integrating a publish-subscribe information management middleware with the Navy's enterprise and tactical platforms, which included applications for information and service discovery, availability monitoring, and asynchronous messaging. We describe the unique challenges presented in the Navy LTE environment and our approaches to address them, which should be applicable to similar military, DIL, and tactical environments and challenges.

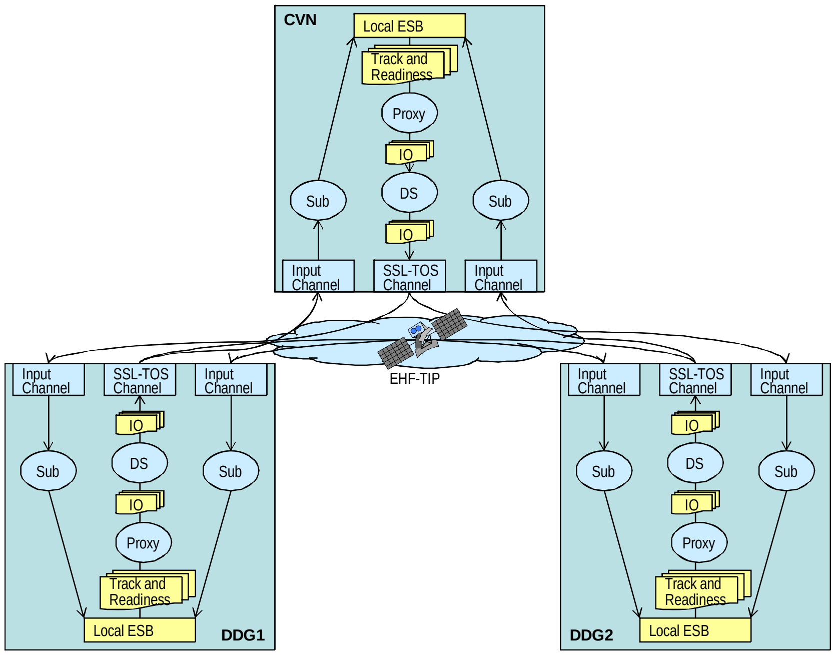

Figure 3. The Navy CS/C2 LTE and Mission Thread 3 Architecture.

2.1 Information Management Challenges

Prior Phoenix+QED research demonstrated [10] the ability to assign policy to information, actors and operations; ensure preferential treatment for high priority information flows; and effectively allocate resources to meet mission needs in a LAN environment. In reporting prior empirical results, the Phoenix+QED IM Services have been deployed as a centralized and system-independent publish-subscribe middleware in a LAN environment. To facilitate controlled experiments and easily quantifiable analysis, we evaluated the system using sample information that lacked the complexity and richness of the data present in the LTE. The blended tactical and enterprise Navy environment from the LTE presents three challenges for IM services such as Phoenix+QED: (1) the use of a DIL WAN, (2) realistic data models with mission policy specifications, and (3) the fluid nature of the LTE integration scenarios, environment, and testbed.

Previous work [4] has highlighted common issues in deploying middleware services over satellite communications. Constrained resources and the dynamic nature of satellite communications present several technical challenges when managing either the dissemination of information or communication interactions between remote services. Network attributes such as extremely long latencies, intermediary communication buffering, varying support for differential types of service, and intermittent connectivity may impact the ability to deliver information in a timely, prioritized, and controlled manner.

The Navy experiment includes two new data models that describe approximately 84 unique, and *Mission Thread Three* relevant, data types in two categories, described in more detail in Section 2.2:

- System Vehicular Track data, which is high rate data such as might be produced by radar systems.
- Readiness data for MH-60R helicopters, produced at a lower rate than track data.

Both data models are rich in attributes, and when fully populated by a data generation tool, individual objects vary in size. The LTE defined a mission prioritization scheme (described in Section 2.2) that describes an operations-driven policy specification for how Track and Readiness Information should be prioritized during dissemination. The data richness, variety in size and type, and the information prioritization scheme for the LTE is more complex than those on which we had previously demonstrated and evaluated QED.

The final challenge in the Naval setting is qualitative in nature. Over the course of several months leading up to, throughout, and following the LTE, a fluid enterprise integration environment brought together several mature and cutting-edge research components to form a large distributed test-bed. Several independent teams from different government and commercial organizations worked side-by-side to complete disjoint milestones for the Mission Threads defined under the LTE's overarching vision. The evolving nature of component integration, and thus the dependent experiment definition, challenged Phoenix+QED to not only stand alone as a QoS-managed pub-sub middleware, but to participate fully as a distributed set of IM Services among several other evolving software components.

2.2 Information and Policies

The Navy's Vehicular Track and MH-60R Readiness data models describe over one hundred information types, of which 84 types were relevant to the scenarios of the LTE. The LTE included IDL models that provide a language independent and structured definition of attributes for vehicular tracks (e.g., classification, vehicular type, and kinematic data) and helicopter (*Helo*) readiness information (e.g., sensor status, weapons inventory, and tasking information). Experiments in the Mission Thread that we are describing, Mission Thread Three, use simulation tools and define run profiles to ensure the reproducible population of data objects. Over a one hour run, we observed fully populated POJOs for track and readiness objects, after IO wrapping and compression for WAN delivery, averaging 12KB and 5260B in serialized sizes respectively. The same object group's variance was 1560B and 1750B.

The Information Priorities Policy for Track Information is shown in Table 1. By extracting specific attribute values from IOs, Phoenix+QED can apply the appropriate priority to information. As indicated in Table 1, information priorities are determined using combinations of classification (Friend, Neutral, Suspect, Hostile, or Unknown), type (Surface, Subsurface, Air, or Unknown), and kinematic information (including whether a tracked vehicle is in a specified Operational Area, OPAREA). Positional information is in the form of latitude and longitude in decimal degrees. The operational area for the experiment refers to a square region bounding the expected position of the three ships and any tracked vehicles. When simulating the movement of tracked objects, experimental run profiles move vehicles in straight line and rosette patterns that enter and leave the operational area over time.

Table 1. Track priorities.

	Surface	Subsurface	Air	Unknown
Friend	2	1	2	N/A
Neutral	3	N/A	3	N/A
Suspect or Hostile	1	1	1	N/A
Unknown or Pending & In the OPAREA	2	1	2	2
Unknown or Pending & Not in the OPAREA	3	1	3	3

In addition to the priority specification, the LTE scenarios specified information preferences for track replacement (i.e., update a queued track of the same globally unique identifier, GUID), and a two minute deadline for air tracks. Mechanisms for both the replacement of in-the-queue information with new information and the enforcement of a deadline for information are supported by QED [14]. QED enforces the replacement policy prior to enqueuing a Track IO for dissemination by performing a linear search on a client's queue. If a matching GUID is located, the Track IO on the queue is replaced with the new Track. Otherwise, the new Track is inserted at the back of the queue. Deadline enforcement occurs in three locations: (1) at the time of initial enqueue for dissemination, (2) prior to a subscribing client queue's priority bin being considered for aggregate dissemination scheduling (i.e., across all of the client queues. QED ensures fairness across connections), and (3) immediately prior to the bandwidth manager's *select-for-send* operation, which occurs when the appropriate amount of bandwidth is available to send an IO.

Table 2 highlights the Information Priorities Policy for Readiness Information, i.e., information about the readiness of various systems on Helicopter platforms (i.e., Radar, Forward Looking Infrared Radar, Weapons, etc.). Priorities are applied to MH-60R information provided by two simulated 'Helos'. The two helos may be in one of two positions (Position 1 or 2), but never at the same time. The priorities listed in the table are only applied when a status change occurs. Unchanged readiness 'updates' are delivered with a default priority of 3. For example, if a radar status has not changed, the priority of that information is 3. In contrast, if a radar status has changed, readiness information about a radar system from a helo in position 2 has a priority of 1 (if the helo is in position 1, the readiness information about the

Table 2. Readiness status change priorities.

	Position 1	Position 2
Radar	3	1
FLIR	3	3
Weapons	1	1
Sonobouy	N/A	2
ESN	N/A	1
Dipping Sonar	1	N/A

radar is still priority 3 even if it has changed).

Assigning the priorities listed in the Information Priority Policy for readiness data (Table 2) requires inspection and comparison of attribute values in the metadata and/or state of the information. This functionality is not directly supported by QED's policy language, although it is related to *context-aware QoS* features we have been researching [13]. In order to perform prioritization under these requirements, we implemented an upstream component, *the readiness parser*, to store and retrieve existing readiness state. The readiness parser, further described in Section 2.4, is responsible for evaluating existing state against prioritization goals and providing a recommendation to QED's policy enforcement mechanisms.

2.3 Platforms and Network Characteristics

The LTE's experiment environment consists of three local area network platforms physically located on the east coast of the United States, and interconnected via long-haul fiber to west-coast satellite terminals for intra-platform communication. Each platform simulates a Navy shipboard environment; specifically a CVN (Aircraft Carrier) and two DDGs (Destroyers). Platforms are constructed using approximately ten virtual machines (VMs) that perform a set of functionalities to produce track and readiness data, operate CS/C2 systems, provide core enterprise services, and centralize event logging. We deploy two Phoenix+QED VMs per platform for publishing information and subscribing to information. Our VMs run Red Hat Enterprise Linux 5 and are allocated two cores with about 2GB of memory.

Each platform hosts four VMs running LTE infrastructure, including the following:

- A *Syslog-ng* server [19], used for collecting the log information used for later analysis. Each mission thread includes *instrumentation points* where instrumentation code collects information such as the number and types of information received and forwarded at the point, and the priority assigned to the information. A centralized Syslog-ng server is used to capture data from multiple time-synchronized VMs for post-execution analysis. Logging data from Phoenix+QED's instrumentation points is relayed over TCP to the Syslog-ng server.
- The Office of Naval Research's *Force Discovery Service (FDS)*, used to discover services that are registered in the Navy LTE system, such as the publishers of track and readiness data.
- *JBoss Operations Network (JON)*, used to monitor and for administration of the registered services. We included a Phoenix+QED JON plugin that registers the outward facing (i.e., visible to the rest of the Navy LTE components) Phoenix+QED components. This aids in resource discovery and availability monitoring of the Phoenix+QED components.
- A *MRG-M AMQP broker* [15]. MRG-M (the Messaging implementation of Red Hat's Messaging/Realtime/Grid software) is an open-source implementation of the *Advanced Message Queuing Protocol (AMQP)* which provides a messaging-based interface to the ESB on each platform. A Phoenix client registers for track and readiness messages from the local ESB through MRG-M, and ultimately transforms received data into an IO for dissemination across the WAN. Once a track or readiness IO is received from a remote platform, the Phoenix subscribing client inserts the ESB payload into the local ESB via MRG-M.

Interactions with the FDS and MRG-M allow Phoenix+QED to receive event updates when new Track and Readiness producers register AMQP topics and to ultimately consume information if a particular topic is of interest.

Inter-platform packets are routed to an EHF-TIP terminal [24]. Geographically, the EHF-TIP terminals are located across the country from our VMs and are accessible via a long haul fiber-optic network. Intermediary routers (i.e., between the local platforms and the TIP terminal) perform active queue management (AQM) to throttle bandwidth and aid in the enforcement of Differentiated Service Code Points (DSCP) set in three TOS bits. The EHF-TIP configuration specifies the CVN terminal as the TDMA controller. A static bandwidth allocation of 128 Kbps is configured for the CVN, while the two DDG terminals are each provided a static allocation of 32 Kbps. The aggregate bandwidth capability of the EHF-TIP network is 512 Kbps, and upon demand any one terminal is capable of dynamically transmitting up to, but not more than, 256 Kbps. In other words, the remaining 320Kbps of bandwidth may be dynamically allocated to terminals on a demand basis. This dynamic allocation of bandwidth by the TDMA controller, which is not visible to or controllable by the application layer, has an unanticipated interaction with the QED QoS management, as described in Section 4.

The QED bandwidth manager enforces a user-defined bandwidth threshold across all of its outbound connections. This open-loop controller allows QED to enforce a broker-wide bandwidth allocation to meet user needs. Considering the static and dynamic characteristics of the EHF-TIP configuration, and assuming an average request load across the three platforms, we set a threshold of 29,375 KB/s (235 Kbps) and 17,375 KB/s (139 Kbps) for the CVN and DDGs, respectively.

While QED's administration GUI may be used to tune the bandwidth manager's allocation at runtime, it is hard for a human-in-the-loop to respond frequently and accurately to resource changes in a DIL network. Several fixed network characteristics and other dynamic factors such as transient faults and competing traffic in the DIL network may affect QED's ability to effectively manage bandwidth. Furthermore, DIL effects at the radio link may result in unpredictable delays during transmission.

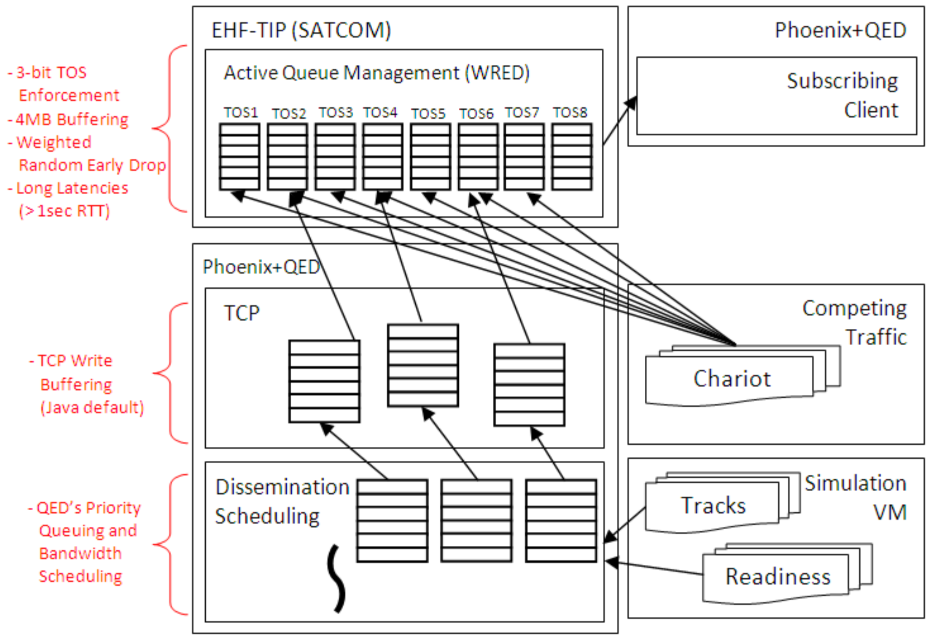
EHF-TIP (SATCOM)
Active Queue Management (WRED)
TOS1 TOS2 TOS3 TOS4 TOS5 TOS6 TOS7 TOS8
- 3-bit TOS Enforcement
- 4MB Buffering
- Weighted Random Early Drop
- Long Latencies (> 1sec RTT)
Phoenix+QED
Subscribing Client
Phoenix+QED
TCP
- TCP Write Buffering (Java default)
Competing Traffic
Chariot
Dissemination Scheduling
- QED's Priority Queuing and Bandwidth Scheduling
Simulation VM
Tracks
Readiness

Figure 4. Factors affecting application level scheduling of outgoing information.

Figure 4 summarizes many of the factors that may impact QED's bandwidth scheduler, which determines the time-to-send using a combination of the object size and available bandwidth. Writes occur over a mutually authenticated SSL connection, and a flush call is performed immediately following the write object call. A flush call on a socket's ObjectOutputStream results in all data being pushed down to the native socket libraries[1]. At this point, control of the message delivery is granted to the network. As shown in Figure 4, there are several details that affect QED's ability to control dissemination, including network features for DSCP support, 4 MB of buffering across eight precedence-levels that exists at the TIP terminal (i.e., 512KB per level), and an AQM implementation of weighted random early drop (WRED). These features offer no cross-layer feedback to support QED's application-level bandwidth manager.

2.4 Phoenix+QED Instrumentation

Figure 5 summarizes the modular components and instrumentation points for the two Phoenix+QED VMs on each platform. Three processes executing across two VMs are responsible for moving track and readiness information between local and remote C2 systems. Several AMQP consumer queues are registered in response to information discovery calls to the Force Discovery Service (not shown). Track and readiness messages are consumed and sent to the appropriate parser, where they are asynchronously queued for evaluation. Each parser uses a thread pool with a configurable number of threads to evaluate information in a First In First Out (FIFO) order. The track parser uses XStream [23] and XPath [21] to evaluate the track against the LTE's Information Priority Policy. The readiness parser examines the public instance members of the JBoss ESB object.

Once a priority for data is determined, the parsers pass the object to the Phoenix Client for packaging as a Phoenix IO, using attributes from the C2 messages to populate the attributes of the IO. Once an IO is packaged, a single-thread dispatcher is used to asynchronously send the IO to the IM services, which in the case of the LTE is primarily the Dissemination Service. At each logging point, a log message is enqueued at a singleton-component that asynchronously delivers all instrumentation messages to the local platform's Syslog-ng server via two-way SSL. Instrumentation points, jointly defined with the Navy's experimentation team, log the following:

[1] The Phoenix+QED prototype is written in Java.

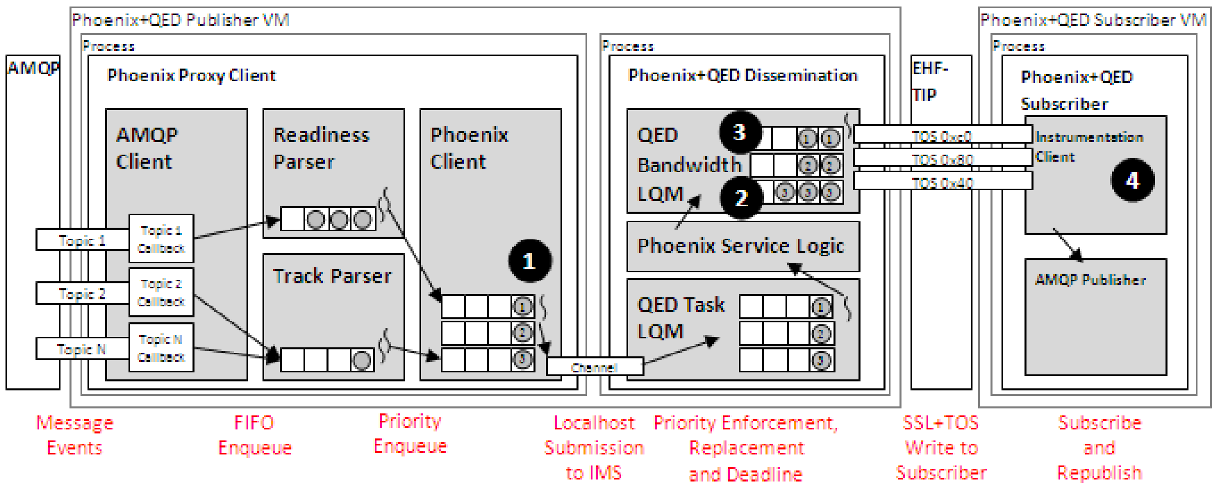

Figure 5. Phoenix+QED instrumentation.

1. All track and readiness information retrieved from the ESB. This establishes the information produced by the local sensors that are available to be disseminated to other platforms.
2. Each IO as it arrives at the Dissemination Service. This records all tracks and readiness IOs that the Dissemination Service could choose to send.
3. Each IO selected for Dissemination, immediately prior to sending it. This records the information that Phoenix+QED attempts to send between platforms.
4. Each IO that is received at the other platform and the time it is received. These are track and readiness IOs that are successfully disseminated.

The Navy LTE infrastructure also includes a logging point recording all tracks and readiness information successfully published to the remote platform's ESB. This instrumentation point, which is not shown in Figure 5, records each successful end-to-end delivery of track and readiness information.

Phoenix and QED Services are deployed as Spring Beans [18]. QED task management controls the execution of application logic, here the Dissemination Service, in order to meet policy and computational resource goals. QED bandwidth management is used to enforce policy decisions and meet communications resource expectations during the dissemination of information.

We use two types of Phoenix Channels for delivering IOs, a Localhost TCP channel for intra-process communication (i.e., Publisher-to-IM Services) and a mutually authenticated SSL channel with type-of-service support for inter-platform communication (i.e., IM Services-to-Subscriber). For each client connection, three SSL channels are used to map the type-of-service priority levels between the IM services and the subscriber. When an IO is sent, instrumentation point 3 sends the precedence level, the client, and information about the object to Syslog. The Dissemination Service also logs messages if a replacement or deadline enforcement action occurs.

3. EXPERIMENTS AND METRICS

Three LTE experiments (shown in Table 3) examine different dimensions of Phoenix and QED, including how well QED performs under both a stress test and network outage scenario, and how well the Phoenix software performs without QoS-management from QED. All experiments run for approximately 45-60 minutes and introduce competing background traffic at a low-rate. Competing traffic is generated through the scripted execution of a traffic generator, *Chariot*, at about 300 Bytes/second across the seven EHF-TIP precedence levels. The intent of the background traffic is to mimic different traffic classes, such as text messaging which is used in fleet settings, to incorporate an additional dimension of realism into the experiment.

The first experiment, *a stepped-stress load experiment*, generated track and readiness data at varying percentages of the total bandwidth available. In ten minute intervals, data is generated at 50%, 100%, 150%, and 250% percent of total EHF-TIP network capacity. The final 5 minutes of the experiment run returns the load to 50% as a cool-down interval.

The stepped-stress experiment is structured to exercise Phoenix+QED's ability to manage the high volume of publisher events and enforce deadline and replacement policies while maximizing the utility of disseminated information over an under-provisioned network.

The second experiment examines the Phoenix software *without* QED using the same stepped-stress load execution framework from the first experiment. This experiment serves both as a baseline (one of two baselines) and as an experimental run. The Navy LTE system without Phoenix+QED (executed as part of one of the other Mission Threads) served as the main baseline for comparison for Mission Thread 3's experiments, and the Phoenix-only results from this second experiment can be compared against that baseline to see how Phoenix's IM services perform without the QED features. In addition, the Phoenix+QED stepped-stress load results can be compared against the Phoenix-only run to evaluate the effects of the QED features on the Phoenix baseline.

The Phoenix-only run disables QED's QoS management logic in the Phoenix Dissemination Service. The experiment uses the same proxy publishing client from the network and QED stress test runs to submit data to Phoenix. As in the other experiments, the Proxy client parses incoming tracks and readiness data to determine a priority for data elements, but assigns a type-of-service precedence level of four to all data to ensure equal treatment on the wire. In this configuration the dissemination operation is not resource or priority aware, and should result in the FIFO dissemination of track and readiness data without regard for the resource limited TIP network.

The third experiment, *a network outage experiment*, introduces controlled failures on intermediate routers residing between the LTE's three platforms. From the experiment onset, traffic is generated at 100% of the network capacity and continues throughout the experiment. Controlled outages of 15, 30, 90, 150, and 600 seconds are introduced throughout the run. Smaller recovery periods are invoked to simulate the chop-in-chop-out feel of a MANET system.

Table 3. Summary of LTE experiments.

No.	Experiment	Simulation/Stimulation	Configuration	Summary of results
1	Stepped Load	Varying loads of Track & Readiness data, with varying priorities (50% network capacity, 100%, 150%, 250%, 50%)	Phoenix, with QED bandwidth control, prioritization on dissemination, setting of packet TOS bits, deadline enforcement, & replacement	• QED appeared to deliver high priority tracks and readiness preferentially • Latencies of high priority tracks were lower • QED kept EHF-TIP from becoming overloaded • QED enforced deadlines and replaced older tracks with newer when appropriate
2	Baseline	Same as Exp. No. 1	Phoenix with QED functionality turned off. Reuses proxy client code.	• All tracks and readiness treated the same; no differentiation • Network overloaded
3	Outage	Start with traffic at 100% of network capacity and induce link failures of varying durations from 15 seconds to 10 minutes.	Same as Exp. No. 1	• Information continued to be propagated in priority order across the connections that were up • When links recovered, Phoenix began sending again with prioritization

Prior to execution, we defined a set of metrics for evaluating Phoenix+QED, shown in Table 4. Metrics covering prioritization accuracy, characteristics of priority-based dissemination, and utility of information as delivered to subscribers were considered. These metrics offer one model for evaluating QoS performance, and consider many of the facets of information-level service quality including timeliness, suitability, accuracy, and the completeness of actions such as information delivery.

Table 4. Experiment metrics.

Metric	Description	Goal
Prioritization Accuracy	Result of Prioritization process	Matched to specification
High Priority Information Loss	Number of dropped high priority messages compared to total dropped messages	0
Priority Ratio of Non-delivered Information	Average priority of all delivered information / Average priority of all non-delivered information	Greater than 1
Latency	As a function of priority: High priority: minimal latency, regardless of available bandwidth (until bandwidth < high priority volume) Low priority: latency inversely proportional to available bandwidth, limited by volume of high priority data	High Priority: 0 Low Priority: Proportional to available bandwidth
Information Timeliness	Number of IOs delivered after deadline plus transmission latency	0
Information Staleness	For information that is marked as replaceable, the number of IOs with an older timestamp arriving after an IO with a newer timestamp	0
Total Information Loss	Volume of data received by consumer/volume of data available, as a function of available bandwidth	Graceful degradation
Recovery Window	After a network outage, time to recover consistency in track and readiness picture across the WAN	Proportional to duration of outage

3.1 Data Generation in the LTE Experiments

Since this paper details our experiences from the eyes and role of a 'performer' in the LTE, many traditional aspects associated with experimentation, such as design and implementation, were completed by the Navy's Analysis Team. Throughout the experiments, specifics such as the type of tracks that are generated, how often readiness information is injected, and how vehicles move throughout the operational area, were not disclosed to performers. Many of these details will be covered in the Navy's final report on the LTE. Table 5 displays what we currently know about how track data was generated by priority during the experiment.

Table 5. How data was generated during the experiments.

	Surface	Subsurface	Air	Unknown
Friend	2	1	2	N/A
Neutral	3	N/A	3	N/A
Suspect or Hostile	1	1	1	N/A
Unknown or Pending & In the OPAREA	2	1	2	2
Unknown or Pending & Not in the OPAREA	3	1	3	3

The highlighted areas in Table 5 represent three types of tracks that were created throughout the runs. Air tracks either moved in a rosette pattern in and out of the OPAREA or across the OPAREA in a straight line. The air track motion resulted in a priority shift between 2 and 3 during the experiment.

4. RESULTS

This section is divided into three subsections covering the QED stress test, the Phoenix stress test, and the QED outage test. The results in this section show several key findings, including the following:

- Phoenix+QED effectively prioritized information and provided appropriate differential service for track and readiness information.
- 36% of all priority 1 track and readiness data was prioritized, enqueued, and disseminated in less than 2 seconds in the stepped-load run.
- Pre-WAN latency distributions show a clear advantage for priority 1 information over priority 2 information.
- Phoenix+QED replaced 81%, 80%, and 74% of older data with newer information on the CVN, DDG, and DDG2 platforms, respectively, during the stepped-load run.
- Counting only payload received, Phoenix+QED achieved an average WAN utilization of 66.1% with a variance of 8% across the stepped-load run, with a peak sustained WAN utilization of 79% of the available bandwidth. Exclud-

ing the first and fifth steps, at full load Phoenix+QED averaged over 70% utilization with a 4% variance.[2]

- Track and readiness data was generated non-deterministically and out of the planned control range during the stepped-load runs.

4.1 Evaluation 1: QED Stress Test

The intention of the QED Stress Test is to present four ten-minute load intervals as steps from 50%, 100%, 150%, and 250% of the total EHF-TIP's capacity. A fifth step returns the load to 50% of capacity and runs for approximately five minutes to clear out any remaining data in the queues. An aggregate visual depiction of the track and readiness data generated during the experiment is presented in Figure 6 for the three platforms. This figure bins AMQP generated message counts into 5-second bins. Spikes in the message counts on the DDG1 and DDG2 data sets are representative of Readiness information injections. During execution, Readiness information from the collective set of simulated MH-60R sensors were injected at two minute intervals for both of the DDG platforms.

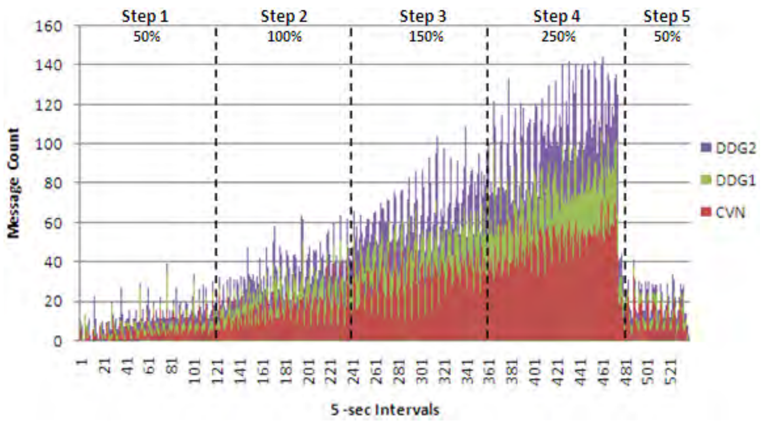

Figure 6. Graph of the stepped up load during experiment 1.

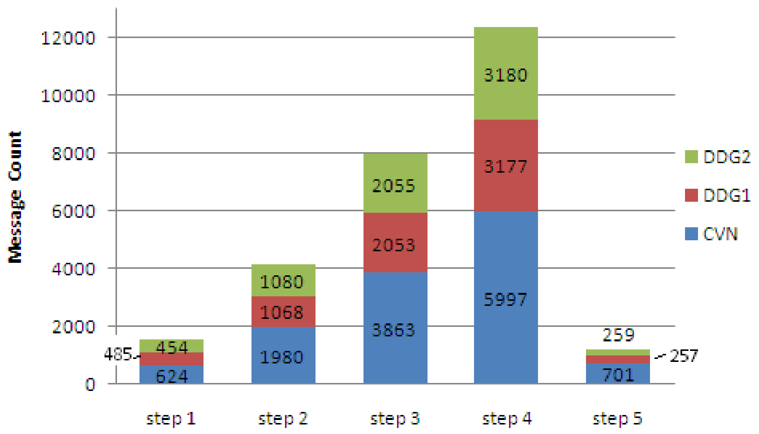

Figure 7. Detailed look at the traffic load during the stepped load experiment.

As a performer, and not privy to the many details of the construction of the experiment harnesses load profile, we did not have a great deal of insight into how load would be produced during the evaluations. Our single insight into how to configure QED's bandwidth thresholds resided in the EHF-TIP network configuration. The WAN was configured with static bandwidth allocations of 128kbps and 32kbps for the CVN and two DDG platforms respectively and with a maximum dynamic cap of 256Kbps for any individual platform at any point in time. Considering this assumption, we observe a static allocation bandwidth ratio of .25-to-.0625-to-.0625 between the platforms, where any single platform may not have more than one-half of the entire bandwidth during the entire experiment. Conservatively, we considered equal loading (i.e., a 1-to-1-to-1 loading ratio) between the three platforms and equally allocated the space of the dynamic bandwidth allocation between the platforms. This resulted in bandwidth caps of 29,375 KB/s and 17,375 KB/s for the CVN and DDGs respectively.

Figure 7 shows a more detailed look at the stepped load. During the actual execution, load was generated in a 2-to-1-to-1 ratio between the CVN and the two DDG platforms. This observation implies that the network supporting the DDG platforms relies *heavily* upon the dynamic TDMA allocations in comparison to the CVN platform.

A second observation from the loading is that the LTE's data generators did not scale exactly to the 50%, 100%, 150%, 250%, 50% proportions as expected. Instead, we see a 269%, 193%, and a 155% increase between steps 2-4, before a large reduction to 10% of step 4's load in the final step. Since the experimental framework and design for loading were outside of our control, we can only speculate as to the cause of the mismatch between the experiment design and execution. The local track/readiness delivery system on each platform (i.e., DDS on a LAN) exhibited some data loss, which

[2] This, of course, omits some necessary amounts of the full information size, including the IO wrapper of the payload used by Phoenix (which is unique to Mission Thread 3, although there might be similar structure overhead in other threads) and transport level overhead, such as TCP overhead (which should be shared in all Mission Threads).

might account for some portions of mismatch. It is also possible that the combination of the load-profile and semantics of the data generation software resulted in non-deterministic or causally-related data generation effects that ultimately skewed the intent of the stepped-loading process. We can envision that varying the types, sizes and attribute population of the source objects might influence the timing required to get data from the generator to the consuming Proxy Client.

Latency spent in the Phoenix+QED middleware. Figure 8 examines the segment of information flow between the AMQP data source through the moment the Dissemination Service makes a decision to send an object over the WAN. The three charts break down the total submission latencies for IOs in the LTE architecture. Total latencies include overhead incurred during queuing, parsing of track and readiness data, prioritization for the LTE priority scheme, and queue management in the Phoenix+QED Dissemination Service.

IOs selected for dissemination meet both the timeliness and suitability policy requirements of the LTE. In this case, timeliness is the 120-second air-track policy that is evaluated prior to dissemination. In order to meet the 120-second requirement for air tracks, the difference between the timestamp at the AMQP-read and the timestamp just prior to dissemination must be less than 120 seconds. The suitability requirement is met by many factors, including the subscribing client's desire to receive the most recent version (i.e., fulfilled by the replacement policy) of any track data.

The three charts in Figure 8 logarithmically scale the in-queue latencies on the Y-axis and plot the time-of-dissemination, in seconds, on the X-axis. Color-and-shape coded plots display a break-down of the IO's priority at the time of dissemination.

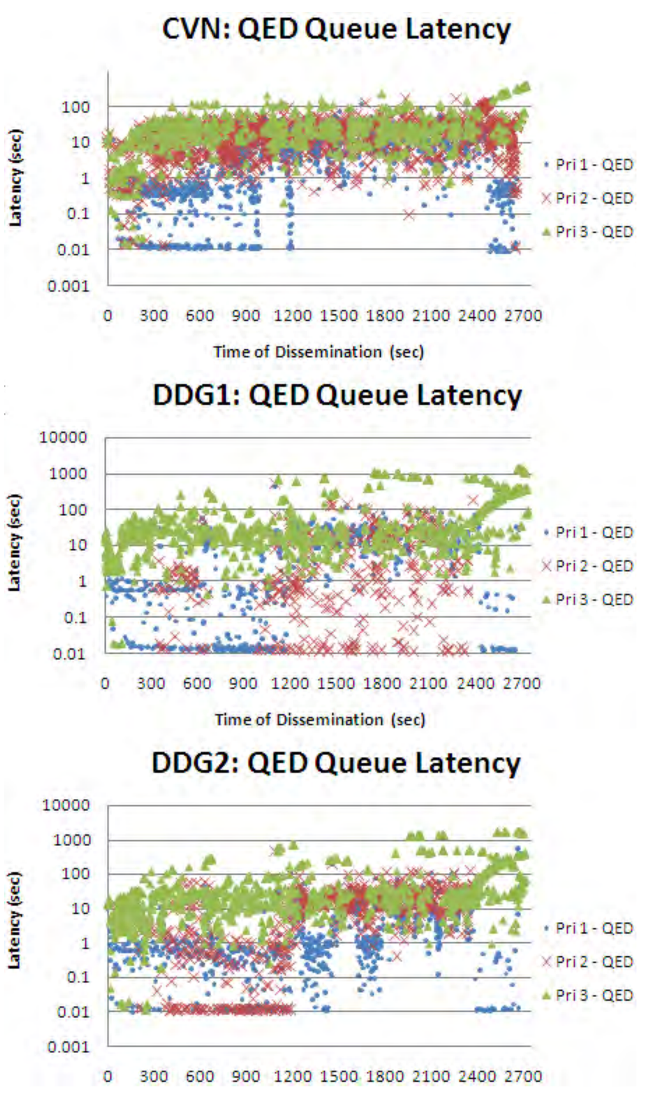

Figure 8. Latency spent in the Phoenix+QED system during Experiment 1.

Working from the top figure to the bottom, we see the expected layering of latencies by priority. The lower priority (priority 3) objects, on a whole, have longer latencies than the higher priority (priority 1) objects. In terms of load, the CVN platform was shown to have generated the greatest volume of information during the experiment. Under this load, the CVN platform visually demonstrates that QED is enforcing prioritization, with an upper bound around the 100 second in-queue latency. In many cases, in step one, two and five we see priority 1 objects with in-queue latencies of less than one second, while priority 2 and 3 objects have latencies greater than one second.

The two DDG charts present similar layering by priority, but also display unusual behavior in response to the DDG loading. As expected, both platforms display the highest latencies for priority 3 IOs, and in several sections, priority 1 objects clearly benefit from the lowest in-queue latencies. Two distinct areas from DDG1 and DDG2 appear to show lower latencies for priority 2 objects. The third and fourth steps on DDG1 (1200-2400 seconds) and midway through the first step and the second step of DDG2 (300-1200 seconds) show some percentage of lower latencies for priority 2 objects compared to priority 1 objects.

Table 6 takes a closer look at step 3 and step 4 on DDG1, by detailing messaging counts and providing latency bins. In both steps, there appears to be an unusually higher percentage of priority 2 objects with low latencies. In step three 73% of all priority 2 objects are delivered in under 10 seconds compared with 18% of all priority 1 objects. Taking a slightly broader view of latencies under 30 seconds, we see that 88-89% of all priority 1 and priority 2 objects fall within this threshold for step three. This pattern is also demonstrated in step four, 50% of all priority 2 objects experience latencies of less than 10 seconds, while 19% of priority 1 objects fall within this category. Once again for this step, a large majority, 79% of priority 1 objects, are scheduled for dissemination in less than 30 seconds.

The table also indicates that in both step three and step four, there were proportionally more priority 1 objects delivered than priority 2 objects. The logging messages that we collected do not have enough detail to fully diagnose what was going on, but we can identify some potential contributing factors. First, the anomaly only occurs on the DDG platforms, not on the CVN, and the DDG platforms publish bursts of Helo data in addition to the track-only data found on the CVN platform. It is possible that a burst of Helo Readiness traffic could affect the traffic pattern of the track data. Second, it is possible that the lower latencies are attributable to the lower volume of priority 2 data, which implies that the separate channels for TOS precedence (i.e., priority delivery) do not work well with the bandwidth scheduling thread under non-uniform loads. Third, we are not able to reproduce these anomalies in our laboratory, even using the configuration and recorded data from the LTE. This implies an unanticipated effect from running in the actual LTE environment which will need to be investigated in future LTE participation.

Table 6. QED latency distributions for step 3 and 4 on DDG1.

	Priority	IO Count	x >= 30 (sec)	30 > x >= 10 (sec)	10 > x >= 5 (sec)	5 > x >= 1 (sec)	x < 1 (sec)
Step 3	1	229	12%	70%	7%	10%	1%
	2	133	13%	16%	7%	20%	46%
	3	192	42%	41%	8%	9%	0%
Step 4	1	213	21%	60%	12%	7%	0%
	2	141	14%	36%	5%	19%	26%
	3	211	49%	35%	12%	4%	0%

Figure 9 examines the distribution of latencies across the three platforms by priority. In this figure, the *y*-axis is logarithmically scaled and displays the message count. The *x*-axis shows individual message counts grouped by priority for latency values between 1 and 30 seconds, a bin for greater than 30 through 120 second latencies, and a bin for latencies greater than 120 seconds. The bins are used to enhance the readability of the figure.

When taken as a whole, large spikes of messages are observed for priority 1 objects with latencies of less than 2 seconds and priority 3 objects greater than 120 seconds. Of the 3987 messages disseminated at priority 1, 1205 messages, or 30% of all messages, were processed and enqueued for less than one second between the AMQP read and the time the IOs were sent over the WAN. When we include the 2 second bin with the priority 1 latency analysis, we observe that 36% of all priority 1 objects were prioritized and routed from the initial AMQP read through dissemination within 2 seconds. At the other extreme, out of 3495 priority 3 messages, 22% or 786 messages were disseminated after in-queue durations of 120 seconds.

In terms of pre-WAN latencies, we believe priority 2 IOs maintained a distinct advantage due to the relative low volume of load at priority 2 levels. From the reported dissemination logging, we know that only 1564 priority 2 messages were disseminated, while both priority 1 and priority 3 IOs had dissemination volumes of over 3400 messages each. This implies that priority 2 IOs, treated equally, had a shorter line to wait in to be serviced. A contributing factor to the volume of priority 2 objects lies within the LTE's prioriti-

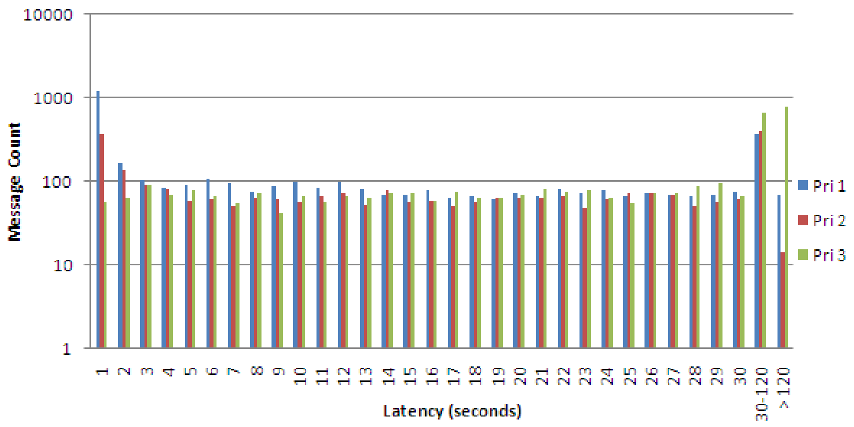

Figure 9. Distribution of latencies across the platforms by priority.

zation scheme. Examining the LTE priority scheme and post-experiment load profiles, we know that only two types of information might have been classified as priority 2: Sonobouy readiness changes, in exactly one configuration, and air tracks inside the operational area that are classified as either unknown or pending. A second contributing factor to the low volume is the 120 second deadline enforcement applied to air tracks. We observed only 14 instances of priority 2 messages delivered with latencies of over 120 seconds, which is consistent with the expected rate of readiness objects (which are not subject to deadline).

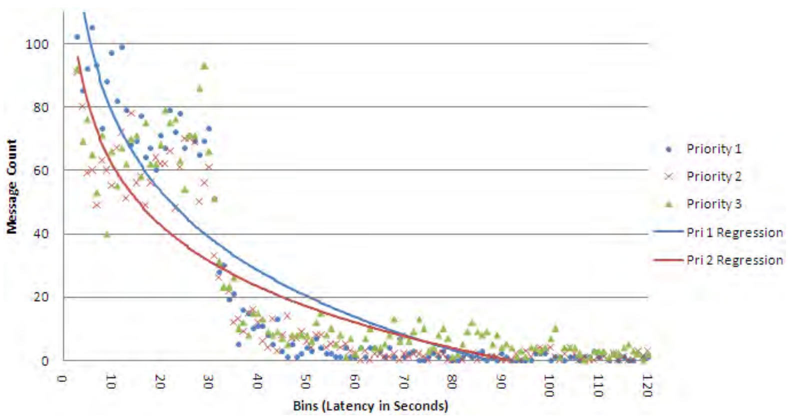

Figure 10. Distribution of latencies across the platforms by priority.

Figure 10 takes a closer look at the distribution of messages with latencies between 3 seconds and 120 seconds for the three priority levels. The figure plots individual latencies for the three priorities. We also plot logarithmic regressions for the priority 1 and priority 2 dissemination actions, which allows us to make additional statements about the relative data points in the two distributions.

Two clusters in Figure 10 immediately jump out: less than 35 second latencies and 35 second through 120 second latencies. In both clusters, we see the behavior we expect. We see higher concentrations of priority 2 and 3 messages at higher latencies. Likewise, we observe much higher message counts for priority 1 objects at lower latencies over this same range. This is evident in the tight clustering of priority one plots (i.e., blue dots) in the upper left portion of the graph. Both trends are positive and indicate that Phoenix+QED responded well to the aggregate load.

The two regression lines attempt to provide a better understanding of how well priority 1 objects faired over the entire set when compared to the priority 2 objects. Examining the regression lines, we see two features that indicate the distribution is speaking to Phoenix+QED's best characteristics. The first is the steep slope of the priority 1 regression compared to the more shallow priority 2 regression between the origin and about the 70 second bin. This trend indicates a greater volume of priority 1 messages exist in the lower-latency bins. The second feature occurs around the 70 second mark. At this point the priority 1 regression drops off (i.e., under the priority 2 regression) faster than the priority 2 regression line. The priority 1 regression reaches the origin of the x-axis around 87 as compared to post-90 for the priority 2 regression line. This implies that towards the second half of the distribution, i.e., at higher latencies, there is a higher count of priority 2 objects than priority 1 objects.

Replacement and deadline enforcement effects on latency. To complete the picture of pre-WAN latencies, we need to review the details of Phoenix+QED's in-queue policy enforcement operations (i.e., replacement and drop) that are used to shape dissemination traffic. Phoenix+QED's drop and replacement mechanisms are used to enforce two of the LTE's policy goals: (i) newer track data of the same track GUID should replace older data and (ii) Air Tracks should not be delivered after 120 seconds. To enforce the replacement policy, for each new track data being enqueued, QED scans the subscribing client's priority queue for a matching track GUID. If a match is found, a replacement will occur. Deadline enforcement is achieved via a drop mechanism which evaluates the submission timestamp and current time at three points in time: (i) during initial enqueue, (ii) at aggregate queuing for dissemination, and (iii) just prior to actual dissemination to see if the 120 second deadline has passed and dispose of the object if it has.

Before reviewing the drop and replacement events in Figure 11, we need to consider the experiment's software topology and our logging points. We know that every track and readiness data read from the AMQP bus will 'fan-out' from a

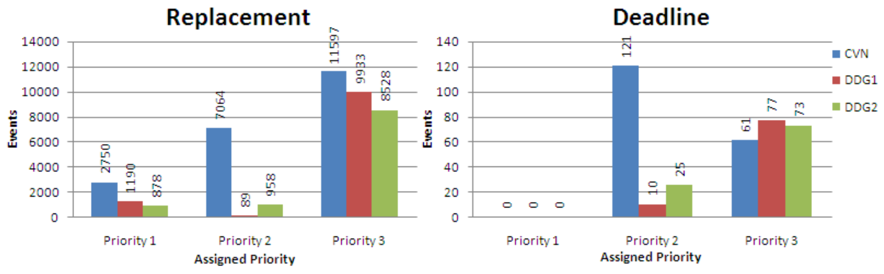

Figure 11. Phoenix+QED queue management events.

single platform's Dissemination Service along the two edges towards subscribers on the other two adjacent platforms. The LTE's full mesh topology has implications when considering the message counts for replacement and deadline events. For every track or readiness data element read from the AMQP-bus, Phoenix+QED is capable of replacing or enforcing a deadline on the fanned-out IO in either one, or both, of the client queues that are responsible for managing the subscribing endpoint. From the AMQP logging points we know that 13,165, 7013, and 7028 messages were submitted for dissemination to CVN, DDG1, and DDG2, respectively. With the full-mesh fan-out we double these message counts to understand how many enqueue operations were processed by Phoenix+QED (i.e., 26,330, 14,026, and 14,056).

Figure 11 shows the number of in-queue replacement events, and the number of deadline enforcement events for the three platforms by priority. In the replacement events chart, by aggregating priorities per-platform, we see that 21,411, 11,212, and 10,364 messages were replaced in total. Taking the fan-out into account (i.e., doubling the aggregate per-platform counts from the dissemination chart), information replacement policies result in replacement rates of 81%, 80%, and 74% of in-queued messages for the CVN, DDG1, and DDG2 platforms, respectively. These aggregate results show that while Phoenix+QED was shaping dissemination operations to the available WAN bandwidth, they were also able to actively manage queue growth. This active queue management ultimately results in the maintenance of the most time-relevant data for subscribers.

In Phoenix+QED's deadline enforcement, we see approximately twice as many deadline enforcements in the CVN platform compared to the two DDG platforms (i.e., 182-to-87-to-98). The spike in priority 2 replacements on the CVN platform may be caused some combination of two events: The greater load on the CVN platform and the experiments runtime population of kinematic metadata on the CVN's air tracks. Since the CVN platform read twice the volume of data from AMQP, it is safe to assume that information's in-queue wait times should, to some degree, be longer on the CVN than the DDG platforms. Another possibility for the 121 priority 2 deadline enforcements may lie in the differences in the generation of kinematic data for air tracks flying through the CVN's operational area. It is possible, and beyond the scope of this paper, that the LTE's load profiles intentionally produced more air tracks in the CVN's operational area where they would be prioritized as 2.

Looking at the CVN's dissemination decisions, we see the volume of information delivery decline as priorities become lower (i.e., from 1-to-2-to-3). Also, the number of replacement events grow as priorities become lower (i.e., 2750, 7064, 11597). This trend is reinforced by earlier latency figures; the longer the duration that higher priority items sit in-queue, the more likely they will be replaced by newer data of the same GUID. On the other hand, the nature of the replacement mechanism is at odds with a strict priority-based interpretation of the reported latencies. Since replaced IO's inherit the newer IO's timestamp, a delivered IO could have a low latency even though it replaced older IOs, whose time in the queue is not included in the delivered IO's latency. It is necessary to consider the degree of replacement when interpreting the priority-based latency as a whole.

The replacement events on the two DDGs present a similar, but slightly different picture compared to the CVN replacement data. Once again, in relationship to priority 1 and 2 information, we see four times the number of replacement events for priority 3 data. Due to what we believe is the lower volume of priority 1 and 2 traffic generated on the two DDG platforms, we do not see similar declines, as compared to the CVN, in numbers of dissemination operations across the priorities. We also do not see linear and upward growth in the number of replacement events as priorities become lower. Neither trend is indicative of a poor performance of the replacement mechanism; in fact, the trends are most likely a response to the AMQP-load imposed upon Phoenix+QED by the experiment.

Latency and throughput in the EHF-TIP satellite WAN. After Phoenix+QED dissemination, the next decomposable component to examine is the EHF-TIP and WAN latencies. Many factors have the potential to affect these latencies, including aggregate traffic volume (i.e., Phoenix+QED information dissemination and background traffic generated by the Chariot software), a collection of intermediate routers moving data between the east and west coast of the United States, the function of the EHF-TIP terminal, and the quality of the satellite link. As stated earlier, the collection of WAN and SATCOM devices support eight differential type-of-service precedence levels. The precedence levels are ordered, from the least important, precedence 0, to the most important precedence 6 for data traffic. Precedence 7 is reserved for control traffic and is used for exchanging routing information. In the following figures, QED's priority 1 (highest) through priority 3 (lowest) importance levels map onto the precedence levels 6, 4, and 2 respectively.

Figure 12 examines WAN latencies by precedence across the steps of the stress tests. In general, and as expected, the average WAN latencies increase as each step increases load, and the higher precedence IOs exhibit lower latency even under extreme load. The only anomalous data point lies in the fifth step, where the precedence 2 average is lower than the precedence 4 latency. Taken as a whole, the average latencies per-step, and in the aggregate category, roughly track the precedence 4 latency. Average latency across the entire test is 18.56 seconds.

The variance figure paints a similar story, but without the uniformity seen across the averages. In general, variance is lower for higher precedence data and greater for lower precedence data, as expected, except in step 5, when the load is reduced and queues of higher precedence information are being flushed at a greater rate. Of interest is the variance smoothing that occurs in step 3 and step 4's overload scenarios. Under the proposed data generation load of 150% and 250% of total capacity, we see the variances stabilize. We believe this speaks to Phoenix+QED's ability to effectively enforce an upper bound on data entering the WAN. One standard deviation across the entire test is 35.07 seconds.

Network utilization. Figure 13 plots a 54 second (i.e., aggregate WAN average plus one deviation) moving average of one second byte bins containing the bytes received across all the experiment's receivers by time. We chose the average latency plus one standard deviation to calculate our moving average in order to capture the majority of observed platform-to-platform average latencies. Both charts in Figure 13 only plot the size of payload in the IOs received. They do not include TCP+SSL protocol overheads that would improve our view of the utilization.

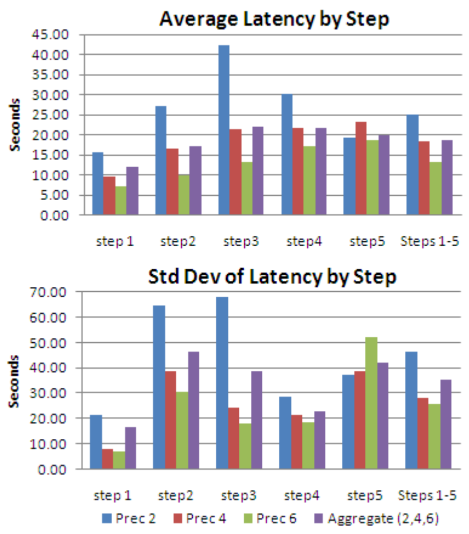

Figure 12. WAN latency across the steps of load by precedence.

The first chart in Figure 13 plots the aggregate platform utilization. Counting only payload received, Phoenix+QED achieved an average WAN utilization of 66.1% with a variance of 8% across the stepped-load run, with a peak sustained WAN utilization of 79% of the available bandwidth. Excluding the first and fifth steps (in which the network was not loaded), at full load Phoenix+QED averaged over 70% utilization with a 4% variance. The actual total utilization of the network is higher than this, because considering the payloads only excludes the amount of the IO wrapper around the payload, bytes added to each packet by the TCP+SSL protocol, and any bandwidth utilized for retransmissions and control messages at the protocol and network layer.

The second chart in Figure 13 plots utilization by precedence level. Excluding the first and fifth step of the stress test run (in which the network was not overloaded), we see that the precedence levels, for the most part, are layered on top of one another as expected. Precedence 6 and precedence 4, the highest two levels, display convex curvature with respect to the x-axis, while precedence 2 (the lowest) is concave with respect to the x-axis. The under-utilized first and fifth steps allow enough room for precedence 2 data, which was over produced during execution, to consume more of the available bandwidth.

4.2 Evaluation 2: Phoenix Stress Test

The second experiment is the evaluation of a Phoenix-only run against the stress test load. The Phoenix-only stress test uses the same five loads and step durations as described in the Phoenix+QED run. This run is intended to serve as a base-line comparison of execution with and without QED's QoS management features. With the exception of minor tim-

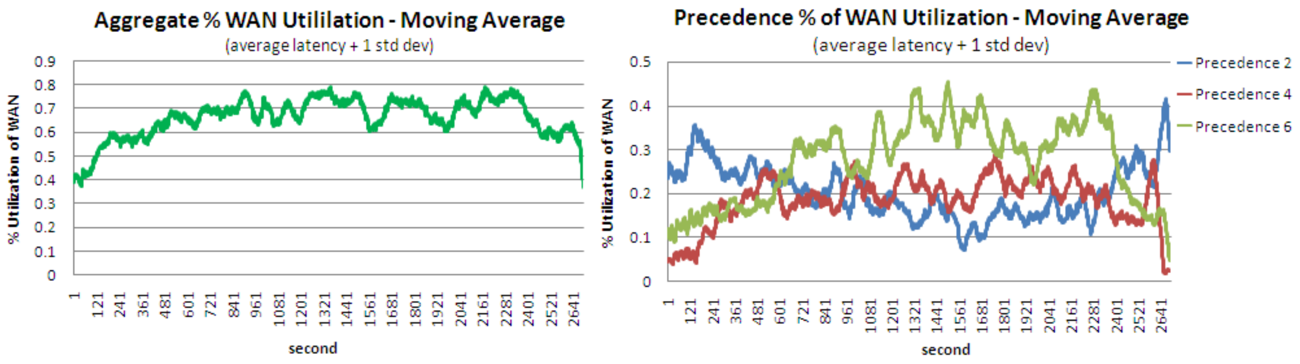

Figure 13. Phoenix+QED's WAN utilization.

ing differences, the execution's load profile looks similar to the results presented in Figure 7 of the Phoenix+QED stress test run.

Figure 14 presents two charts which examine the in-queue and on-the-wan latencies observed during the experiment. The top chart, Phoenix In-Queue Latency, describes the in-phoenix latencies from the time data is read off the AMQP bus through the time data is disseminated over the WAN. The In-Queue latencies are plotted against the time of submission. The bottom chart, Phoenix WAN Latency, describes the over-the-wire time as observed by the time synchronized receivers. The Phoenix WAN Latency chart plots the latencies against the actual time of dissemination. Both charts logarithmically scale the y-axis.

From the In-Queue Latency chart, we observe unsustainable growth in latencies. From the experiment plan we know that through 600 seconds the platforms should only be loaded to 50% of the total EHF-TIP network capacity. Starting around 300 seconds, we observe the beginnings of unsustainable growth on all three platforms. It appears that even before the second step is reached (i.e., at 600 seconds – 100% loading) in-queue latencies exceed 100 seconds. Recall that, based on the LTE policy, air tracks over 120 seconds are of no value to receivers.

With the exception of a small grouping of CVN data points around 1500 seconds on the In-Queue chart, all track and readiness data submitted to the CVN platform from the first step of loading appears to make it through dissemination, albeit with latencies reaching an in-queue maximum latency of 24 minutes (1447 seconds)! It also appears that no data from the second (100% loading), fourth (250% loading), or fifth (50% loading) steps of the CVN load made it through to receivers during the experiment. We observe a more gradual latency growth on the two DDG platforms, and believe the more gradual latency growth is in response to the lower load generated on these platforms. In general, it appears that the unsustainable in-queue latency growth is a function of the number and size of IOs submitted to Phoenix and the blocking nature of the TCP write thread in the Dissemination Service. Once the TCP buffer fills up in response to the under provisioned EHF-TIP network, the write thread (i.e., the disseminate operation of the Dissemination Service) blocks until enough buffer is available to contain the full IO.

The bottom chart in Figure 14 shows the over-the-wire latencies for the dissemination actions during the experiment. Since all IOs are uniformly sent with precedence 4, and there were no restriction of dissemination actions by Phoenix's Dissemination Service, we observe a fairly uniform distribution of latencies across the five steps of the run and across the platforms. This pattern is representative of a best-effort attempt at dissemination, i.e., no preferential treatment of information.

The WAN Latency chart displays an anomaly around 1700 seconds (between the third to fourth steps). It appears that an unplanned DIL event occurred and results in DDG2 being able to send data to CVN and DDG1, but CVN and DDG1 end up in a blocked state. The blocked states are represented by the higher latency blue dot(s) and red crosses at 1650 and 1850 seconds. The latencies of these states are for the most part equivalent to the duration of the unplanned outage. The logs from the experiment do not provide enough information to diagnose this anomaly and the cause requires further investigation.

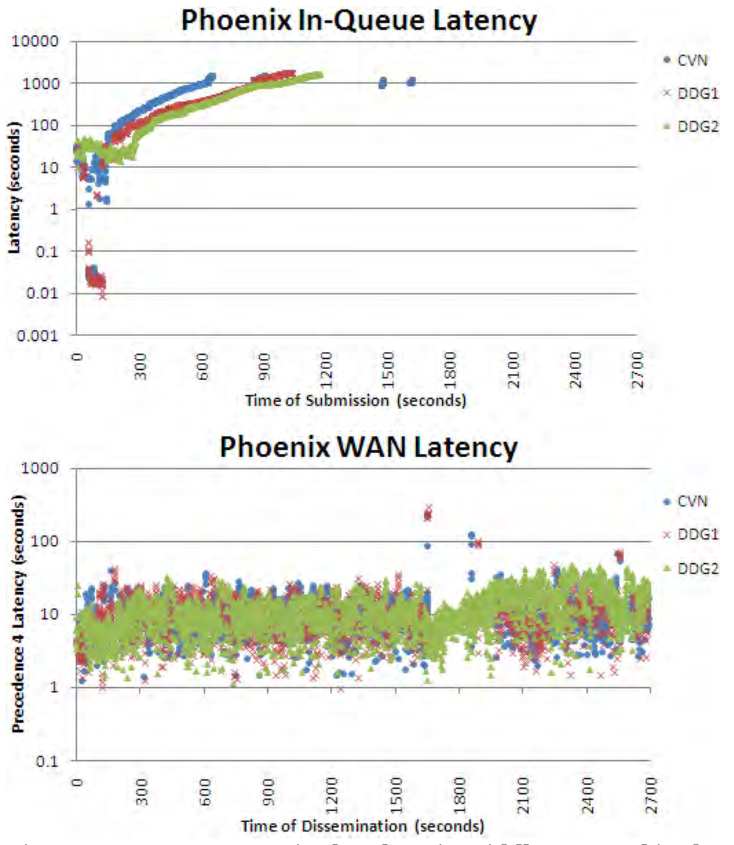

Figure 14. Latency spent in the Phoenix middleware and in the WAN during the Phoenix stress test.

4.3 Evaluation 3: QED Outage Scenario

The third experiment evaluated the effects of controlled linked outages on Phoenix+QED's application-level prioritization. The experiment was designed to help determine the recovery window for Phoenix+QED after a network outage. The experiment generated a volume of data across the three platforms equal to 100% of the total capacity of the EHF-TIP network. Within the WAN, downstream routers between the Phoenix+QED virtual machines and the TIP terminals on the west coast of the United States were disabled at fixed times for varying durations to simulate a chop-in-chop-out effect between the three platforms.

Figure 15 displays the planned outages from the experiment. Red bars in the figure represent durations of outages for the three platforms. The X-axis in the figure is the time from the beginning of the experiment. As described in Section 3, outage occurred for 30, 90, 150, or 600 seconds at fixed times. Outages overlapped two times during the ten minute outage of the CVN (i.e., DDG1 and DDG2 experienced 30 second outages) and effectively left the network in a fully disconnected state.

During the outage periods, we expected several things to occur. First, the Phoenix+QED TCP connections with data on the buffer should end up in a retry state, possibly timing out, and ultimately resulting in a disconnected socket at the application layer. For Phoenix+QED, a disconnection of the socket would result in the execution of a retry connect and a back-off disruption tolerance (DTN) mechanism in Phoenix+QED until the network was healed. Second, Phoenix+QED's bandwidth manager would continue to trigger Information writes until the TCP write buffer was saturated. A saturated TCP write buffer should result in a blocking write operation in the Phoenix+QED thread attempting the write. Third, the outage should alter the EHF-TIP's dynamic bandwidth allocation. In this third case, we believed dynamic bandwidth changes would alter the rate and quality of information flow between platforms, e.g., underprivileged links (such as DDG1-to-DDG2) could receive up to the full 256kbps if they continually populated the TIP buffers when the CVN node was disconnected.

Figure 16 overlays the over-the-WAN information latencies by priority for the two DDG platforms receiving information from the CVN platform over the planned outage intervals for the CVN's down-stream router. The x-axis provides the time of dissemination for an IO (adjusted for time-since-epoch), while the y-axis logarithmically scales the over-the-WAN latency in seconds. The alignment of the outage shading in the figure is equal to the exact execution times from the control plan of the experiment.

Figure 16 shows the two DDG platforms receiving a large amount of data during the planned outage. There are also large gaps in latency data that are offset from the planned outage shadings, but appear approximately the same size. Before assuming a deviation in the experiment control, we consider two factors. First, we take into consideration the over-the-WAN latencies for the data points in the planned outage range. Adjusting for the magnitude of these latencies we see

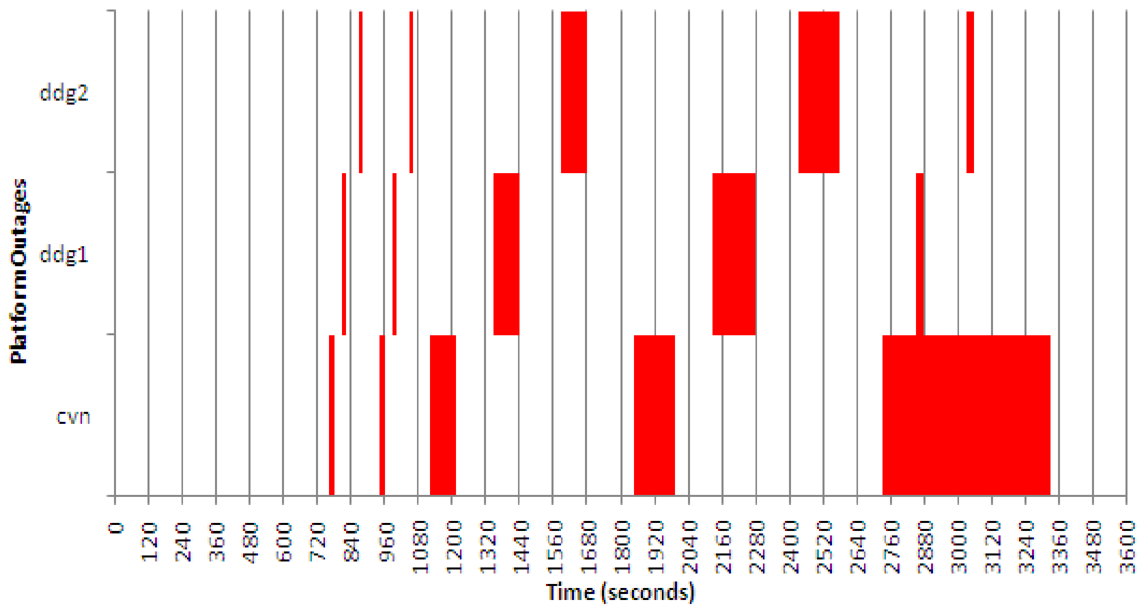

Figure 15. Planned outages by platform.

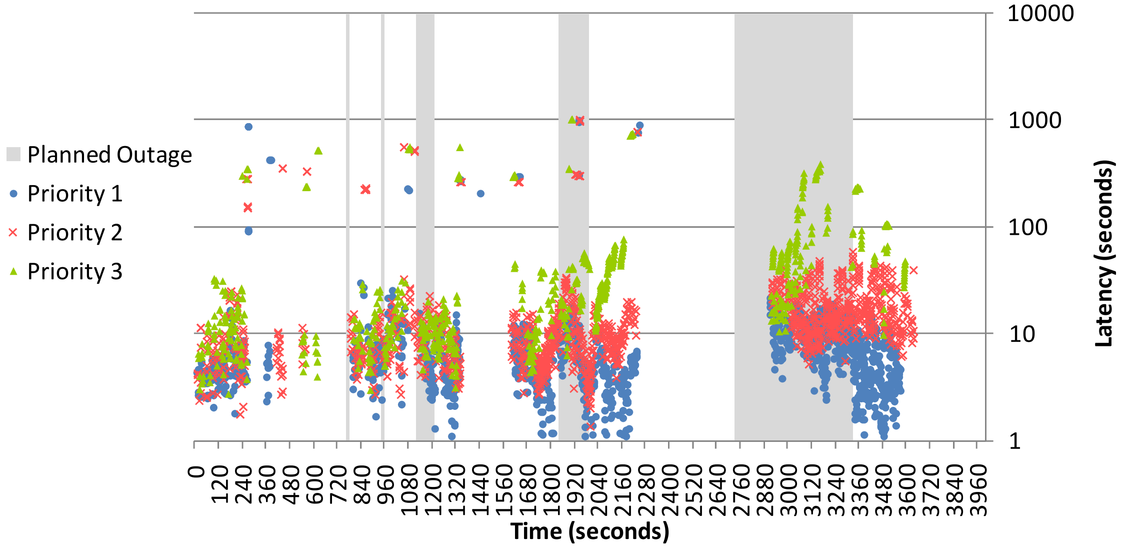

Figure 16. WAN latencies for IOs sent from the CVN by priority during the outage experiment. The shaded bars indicate the occurrence of an outage.

that information still appears to be received in volume when the CVN's downstream router was supposedly offline. Second, we considered the degree of buffered TIP data during the outage. It is easy to conclude that the volume of data (i.e., by just examining message counts) in this figure is much greater than the WAN buffer (4MB).

After looking at individual data points from the CVN's dataset, we believe the planned outages occurred approximately six minutes prior to the execution time's defined control plan. Since Phoenix+QED does not a-priori take into consideration any notion of planned outages, the difference in execution from the control plan has no bearing on Phoenix+QED's overall performance. Using individual data points from the CVN's dataset, we adjust the ten minute outage window for recovery analysis as shown in Figure 17. Taking into consideration the recovery window, we see that the adjustment is almost an exact match and most likely reflects the actual outage during the LTE execution.

Figure 17 plots exactly two objects of each priority before the outage and after the recovery. It also plots all of the disrupted objects (per-priority) that were affected during the CVN's ten minute outage. The objects affected by the outage are clustered tightly in the upper-left quadrant of Figure 17, at approximately 2220 seconds into the run and with over-the-WAN reception latencies averaging 757 seconds across the three priority levels. We shade the area representing the likely actual outage as the time just after the last known successful send followed by 600 seconds of outage.

We plot the recovery window for each priority using the difference between the latency of the last successfully sent object (at the particular priority) before the outage and the latency of the first successfully sent object (at the priority) after the outage. This difference equals a 67 second recovery window for priority 1, an 85 second recovery window for priority 2, and a 147 second recovery window for priority 3. We plot the recovery windows as vertical shaded areas in Figure 17. Examining the data set we observe that the post recovery window latencies return to near run averages for the respective priorities.

The outage scenarios and recovery window analysis for the other two platforms and outage windows are similar to the CVN's analysis. From the CVN analysis, we observe the following insights into Phoenix+QED's recovery from network outage:

- Just prior to the outage and at recovery we observed several positive trends. Prior to the outage, the final objects to be delivered were two priority 1 objects, followed by two priority 2 objects, and lastly the priority 3 objects (stacked on top of each other around 2160 seconds). At recovery time, priority 1 objects recovered the quickest. In a slight visual anomaly, priority 3 objects reach pre-outage latencies earlier in the epoch than priority 2 objects, but when considering their last known (pre-outage) good latency, the priority 3 recovery window is actually larger than priority 2's window.

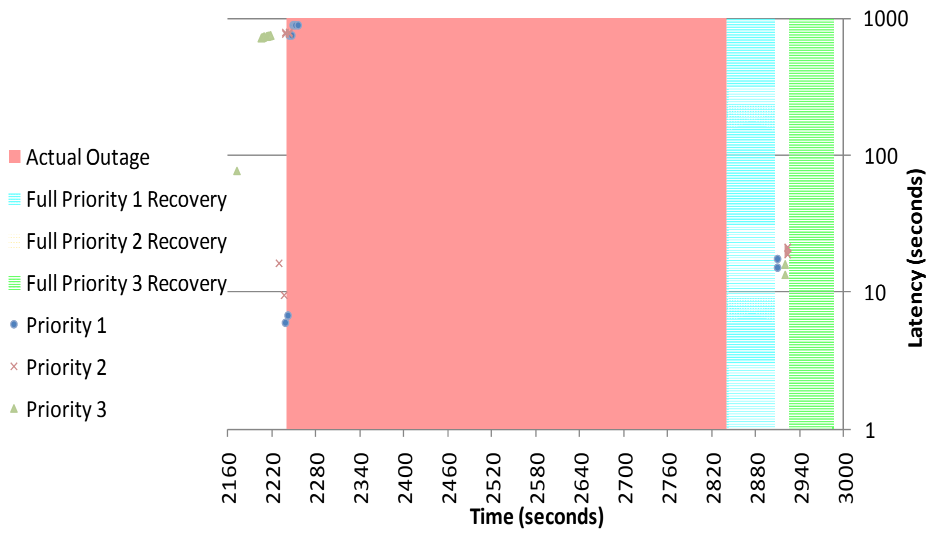

Figure 17. CVN outage and time of recovery.

- During the entire outage experiment run, we did not observe any TCP connection failures. Instead, the Phoenix+QED's write threads appear to have entered a blocked state once the TCP write buffer filled. This trend was apparent in the 'cap' on the number of long latencies observed in Figure 17 (upper left of the figure in a tight cluster). We see this blocking state as a good state for Phoenix+QED. When an outage occurs, Phoenix+QED replacement and drop mechanisms are constantly performing active queue management to ensure that the next round of dissemination when the network recovers will be the most timely and relevant to current operations.
- During the experiment, full recovery from an outage (i.e., the time when an average latency object was received after connectivity was restored) will occur only after the write buffer of the TCP connection is flushed. For the most part, this data is stale data that may not be of much use to consumers. In this experiment 7, 5, and 19 IOs were written into the TCP write buffers for priority 1, 2, and 3 respectively. Priority 3 objects were the smallest tracks during the outage leading to the variance in number of objects buffered. It appears that the volume of bytes submitted for TCP writes were approximately equal across the priorities. If an appropriate network probe or feedback were available, QED could detect the backlog and break the connection to alter the recovery window.

5. RELATED WORK

QoS in the networking community has been defined in terms of traffic engineering, based on a standard definition produced by the International Telecommunication Union [8]. This has frequently meant either resource reservation [25] or differentiated services [7], such as that offered by the EHF-TIP satellite communications in this experiment.

A drawback to providing QoS only at the network layer is that it doesn't translate to user level QoS, i.e., the perceived state of *mission effectiveness* or *user satisfaction* [12]. Network level QoS optimizes for network utilization and packet prioritization, but does not have visibility into the application level semantics of packets. Delaying or dropping an individual packet might make sense from a network level QoS point of view, but affects more than the individual packet, i.e., it affects all other packets making up the information element.

Other approaches to providing QoS interfaces to applications have focused on distribution middleware. Synergy [17] describes a distributed stream processing middleware that provides QoS to data streams in real time by efficient reuse of data streams and processing components. The Data Distribution Service (DDS) is an OMG standard with a rich set of QoS parameters [16], which is implemented in several commercially and openly available implementations. DDS is connection-based and, while it enforces QoS for a connection, it does not provide the aggregate QoS management across connections that share bandwidth that QED provides. Other messaging middleware including the Java Message Service (JMS) [6] and the AMQP messaging service [20] used in the LTE offer some QoS parameters that can influence the delivery characteristics of disseminated information, but do not offer the rich client- and user-level policies that QED provides nor the aggregation over multiple connections. A comparison of JMS and DDS, focusing on their QoS features, is provided in [2].

6. CONCLUSIONS AND LESSONS LEARNED

After performing the research and development to create the QED prototype QoS management software and integrate it with the Phoenix IM services, the participation in the LTE described in this paper gave us the chance to try it out in a realistic environment and set of scenarios different from any we had seen before. We had already performed a significant amount of laboratory experiments and had parts of QED operating in embedded platforms [3], so we anticipated that it would provide more QoS management and control than the baseline system without any QoS management.

For the most part, QED performed very well. In general, it provided overall lower latency and higher throughput to higher importance information. QED managed the flow of information into the network, keeping it from becoming overloaded even under severe amounts of traffic. QED managed network outages well and recovered from outages of a few seconds to several minutes.

One of the lessons learned from the LTE is that a significant amount of integration work didn't go into integrating QED itself into the Navy testbed. Instead the bulk of the integration effort went into the development of supporting infrastructure for the experiment itself, e.g., insertion of logging, interfacing with the Java Operations Network used for service configuration and control, and other infrastructure needed not to support the Phoenix+QED operation, but the operation and conducting of the experiment itself. In addition, we approached the LTE expecting to utilize most of the Phoenix+QED components and ended up integrating and utilizing only a few of the components and features because of the limited scope of the experiments and scenarios. This worked out well for risk reduction and the success of this LTE, and provides an opportunity to move forward from this baseline to experiment with additional Phoenix+QED components in future LTEs.

Another lesson learned is the challenges associated with providing real-time QoS in an environment like the LTE, in which there are several layers of control, but little runtime visibility and feedback for coordinating the control. Specifically, the LTE had three separate layers of dynamic QoS control:

- The EHF-TIP priority queues that enforce TOS bits representing the precedence levels of packets.
- The dynamic TDMA bandwidth allocation that provides more bandwidth to connections with higher amounts of traffic.
- The Phoenix+QED dissemination service that enforces deadlines, replaces stale information, and prioritizes IOs, all at the application layer.

Although we coordinated the layers of control to a certain extent, e.g., by setting the TOS precedence levels for packets consistent with the Phoenix+QED IO importance and by limiting the rate so as to not exceed the amount of bandwidth, each layer of control was independent. That is, there was no centralized control of the layers. Furthermore, there was no visibility API to enable coordination between the layers, i.e., there was no runtime feedback signal to tell QED how loaded the EHF-TIP queues were or how much bandwidth had been allocated (or was available). In the former case, the integrated system worked reasonably well, with Phoenix+QED managing to keep the EHF-TIP queues full, but not overloaded (i.e., we observed no packet drops in the EHF-TIP queue management). In the latter case, however, the QED bandwidth management and dynamic bandwidth allocation interaction led to the anomalies that we report in Section 4. There was no available runtime measure of available bandwidth, the size of queues in EHF-TIP, or the amount of bandwidth dynamically allocated by the dynamic TDMA bandwidth allocation. QED attempted to measure the amount of bandwidth being used by recognizing congestion's effects on TCP acknowledgement. However, this strategy had an unforeseen effect. When QED recognized a delay in TCP acknowledgement, it interpreted it as lower available bandwidth and throttled back the rate of sending along that connection. The reduced rate of sending was then interpreted by the TDMA controller as lower demand and the excess dynamically allocated bandwidth was provided to other connections. In effect, the two conspired to do the opposite of what each was attempting to do individually. In future integrations of these two technologies, we need an API to get information about bandwidth allocation from the TDMA controller and to influence its decisions by QED.

Another side effect of the lack of accurate bandwidth measurement (a common problem in deployed systems) is that Phoenix+QED under-utilized the available network bandwidth. We don't have accurate measurements of how much we under-utilized it, since the bandwidth utilization measurements do not include the IO Wrapper, TCP, and IP overhead. Since reliable runtime bandwidth measurements are generally not available, this is ordinarily handled with testing runs to tune a system. Because of the aggressive LTE schedule and large numbers of experimental "record" runs planned, there was limited opportunity to do this sort of tuning prior to execution of the LTE.

This last point is an important conclusion. Although the integration of Phoenix+QED into the Navy LTE was generally successful and provided higher QoS than the baseline system, the Navy LTE was a one-shot deal. It afforded little opportunity to examine where the system performed well and where it didn't, and use those results to tune, enhance, or configure the system. There were several things that came out of this endeavor that will influence our future designs, integrations, and experiments, and we believe that Phoenix+QED performed well enough to indicate its suitability as a basis for these future efforts.

REFERENCES

[1] Combs, V., Hillman, R., Muccio, M., and McKeel, R., "Joint battlespace infosphere: information management within a C2 enterprise," Proc. Tenth Int. Command and Control Technology Symp., (2005).

[2] Corsaro, A., Querzoni, L., Scipioni, S., Piergiovanni, S., and Virgillito, A., "Quality of service in publish/subscribe middleware," [Global Data Management], IOS Press, (2006).

[3] Gillen, M., Loyall, J., and Sterling, J., "Dynamic quality of service management for multicast tactical communications," Proc. 14th IEEE Computer Society Symposium on Object/component/service-oriented Real-Time distributed Computing (ISORC), Newport Beach, California, (2011).

[4] Gordon, S., and Billington, J., "Middleware services over satellite networks: a survey of issues," Proc. 8th International Aerospace Congress incorporating the 12th National Space Engineering Symposium, Adelaide, Australia, (1999).

[5] Grant, R., Combs, V., Hanna, J., Lipa, B., and Reilly, J., "Phoenix: SOA based information management services," Proc. of the SPIE Defense Transformation and Net-Centric Systems Conference, Orlando, Fl, (2009).

[6] Hapner, M., Burridge, R., Sharma, R., Fialli, J., and Stout, K., "Java message service version 1.1," Sun Microsystems, http://java.sun.com/products/jms/, (2002).

[7] IETF, An Architecture for Differentiated Services, http://www.ietf.org/rfc/rfc2475.txt.

[8] International Telecommunication Union (ITU-T), "Data networks and open system communications, open distributed processing, information technology – open distributed processing – reference model: foundations," ITU-T Recommendation X.902, (2009).

[9] Loyall, J., Carvalho, M., Martignoni III, A., Schmidt, D., Sinclair, A., Gillen, M., Edmondson, J., Bunch, L., and Corman, D., "QoS enabled dissemination of managed information objects in a publish-subscribe-query information broker," Proc. SPIE Conference on Defense Transformation and Net-Centric Systems, (2009).

[10] Loyall, J., Gillen, M., Paulos, A., Bunch, L., Carvalho, M., Edmondson, J., Schmidt, D., and Martignoni III, A., "Dynamic policy-driven quality of service in service-oriented information management systems," *Software: Practice and Experience*, (2011).

[11] Loyall, J., Gillen, M., Paulos, A., Edmondson, J., Varshneya, P., Schmidt, D., Bunch, L., Carvalho, M., and Martignoni III, A., "Dynamic policy-driven quality of service in service-oriented systems," Proc. 13th IEEE Computer Society Symposium on Object/component/service-oriented Real-Time distributed Computing (ISORC), Carmona, Spain, (2010).

[12] Loyall, J., and Schantz, R., "Dynamic QoS management in distributed real-time embedded systems," [Handbook of Real-Time and Embedded Systems] Insup Lee, Joe Leung, Sang Son (Editors), CRC Press, (2008).

[13] Loyall, J., and Schantz, R., "Using context awareness to improve quality of information retrieval in pervasive computing," Seventh IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems, Newport Beach, California, (2009).

[14] Loyall, J., Sinclair, A., Gillen, M., Carvalho, M., Bunch, L., Martignoni III, A., and Marcon, M., "Quality of service in US Air Force information management systems," Proc. of the Military Communications Conference (MILCOM), (2009).

[15] "Redhat.com | MRG," *Redhat.com | The World's Open Source Leader*. Web. 31 Mar. 2011. <http://www.redhat.com/mrg/>.

[16] Object Management Group. "Data Distribution Service for Real-time Systems, Version 1.2," OMG Specification, formal/07-01-01, http://www.omg.org/cgi-bin/doc?formal/07-01-01, (2007).

[17] Repantis, T., Gu, X., and Kalogeraki, V., "Synergy: sharing-aware component composition for distributed stream processing systems," Proc. Int. Middleware Conference, Melbourne, Australia, (2006).

[18] Springsource Community, Spring, http://www.springsource.org/.

[19] "Syslog-ng - Multiplatform Syslog Server and Logging Daemon," *Log Management & Network Security Solutions | BalaBit IT Security*. Web. 31 Mar. 2011. <http://www.balabit.com/network-security/syslog-ng>.
[20] Vinoski, S., "Advanced message queuing protocol," *IEEE Internet Computing* 10: 87–89, (2006).
[21] World Wide Web Consortium. XML path language (XPath) version 1.0. W3C Recommendation, (1999). http://www.w3.org/TR/xpath.
[22] World Wide Web Consortium. XQuery 1.0: an XML query language. W3C Recommendation, (2007). http://www.w3.org/TR/xquery.
[23] *XStream - About XStream*. Web. 31 Mar. 2011. <http://xstream.codehaus.org/>.
[24] Youm, W., Acevedo, M., Heaberlin, A., Chase, D., "Modeling and simulation to support the development of the Navy's Extremely High Frequency TDMA Interface Processor (EHF-TIP)," Proc. of the Military Communications Conference (MILCOM), Atlantic City, NJ, (2005).
[25] Zhang, L., Deering, S., Estrin, D., Shenker, S., and Zappala, D., "RSVP: a new resource reservation protocol," *IEEE Network*, (1993).